debian, os, serveur, install, stretch, scripts, grub, kernel, aptitude, moo

Les branches de Debian

C'est l'une des notions perturbantes lorsque l'on découvre Debian: il existe simultanément plusieurs *versions*, ou plutôt *branches*, en cours de validité. Les plus connues sont **« stable »**, **« testing »** et **« unstable »** mais il existe aussi « oldstable » et « experimental ». Chaque « stable » porte un numéro de version et un nom de code, la branche « testing » ne reçoit qu'un nom de code (le numéro ne sera officiellement attribué qu'au moment ou elle deviendra la nouvelle « stable »). Chacune (sauf la branche « experimental ») reçoit un nom de code tiré des noms des personnages de la saga Toy Story.

- La branche ***stable***

C'est la version officielle de Debian. Figée, elle reçoit des mises à jour encore fréquentes (principalement de sécurité) et apporte la certitude d'avoir un système fonctionnel et stable.
A l'heure actuelle, la version stable en cours porte le numéro 9 et le nom de code **« Stretch »** (la pieuvre du 3e opus).

- La branche ***testing***

C'est la prochaine version stable. Extraite directement de la branche « unstable » à un instant T, elle reçoit des mises à jour de sécurité, des correctifs des paquets existants ou de nouveaux paquets reconnus comme étant matures.
La branche testing actuelle porte le nom de code **« Buster »** (le chien de la famille).

- La branche ***unstable***

En évolution constante, cette branche peut recevoir n'importe quel ajout ou modification à n'importe quel moment. C'est le terrain de jeu favori des développeurs mais aussi la seule branche à suivre de près les évolutions des logiciels. En étant prudent sur les mises à jour que l'on applique ou non, elle est tout à fait utilisable au quotidien si l'on est prêt à être confronté à quelques bugs de temps à autres (mais les développeurs étant extrêmement nombreux et actifs, cela ne dure généralement pas longtemps).
Cette branche se nomme et se nommera toujours **« Sid »** (le garçon sadique avec les jouets mais aussi "*Still In Development*").

- La branche ***oldstable***

Comme son nom l'indique, il s'agit de la précédente « stable ». Des mises à jour de sécurité sont proposée pendant un an, à moins qu'une nouvelle « stable » apparaisse entre temps (cas peu probable mais possible). Au delà, la version est envoyée aux oubliettes et archivée.
Actuellement, c'est la version 8 nommée **« Jessie »** (la cow-girl).

- La branche ***experimental***

Bien qu'elle ne leur soit pas réservée, cette branche est plutôt dédiée aux développeurs. Certains y piochent parfois quelques paquets mais c'est une pratique délicate réservée aux utilisateurs expérimentés.
Cette branche ne porte aucun nom de code.

La rotation des versions

Lorsque la branche *testing* en cours est jugée suffisamment mature, elle est “gelée”. Pendant cette période, plus aucun ajout n'est possible et les seules évolutions concernent la correction de bugs déclarés. Une fois qu'elle est prête à devenir la nouvelle branche *stable*, elle reçoit un numéro de version officielle tandis que l'ancienne *stable* est reléguée au rang de *oldstable* et qu'une copie de la branche *unstable* deviens *testing* et reçoit son nom de code.
Officiellement, il n'y a pas de durée de vie maximale pour une *stable* de Debian puisqu'elle ne sera remplacée que lorsque la *testing* sera jugée prête (et théoriquement, ce serait ainsi même si une *testing* avait besoin de plusieurs décennies pour être jugée prête).
Dans la pratique, l'historique des versions montre que les dernières versions *stable* ont eu une durée de vie d'environ 2ans.

Quelle branche choisir

- Si la robustesse est primordiale, le choix est simple: *stable*
- Pour une machine multimédia ou s'il est nécessaire d'accéder à certaines versions récentes de logiciels spécifiques, il faudra s'orienter vers la branche *unstable* mais cela demande du sérieux et de la rigueur dans la maintenance du système (il faut **impérativement** installer le paquet *apt-listbugs* qui permet de lister les bugs connus et peser attentivement le pour et le contre à chaque mise à jour)
- La branche *testing* n'est à priori une option intéressante que si la *stable* actuelle est déjà âgée (disons au delà de 16 mois, mais chacun fixera sa limite) puisque dans ce cas, la majorité des bugs restant sont mineurs
- Les débutants devraient plutôt commencer avec la branche *stable*, d'autant qu'il est possible d'installer des paquets en provenance de branches plus évolutives ou même directement depuis les sources (c'est très formateur avant de se frotter à une *unstable* pure)

Il est facile de passer de *stable* à *testing* ou *unstable* mais pas l'inverse donc en cas de doute, choisir la branche *stable*.

De quelle branche parle-t-on ici?

La première mouture réellement aboutie de mon serveur jouait bien plus de rôles que la version actuelle et en particulier celui de routeur WiFi. C'était environ un an après la sortie de la version 5 de Debian qui comptait à ce moment là un énorme retard sur ce type de techno et la version stable était, à l'époque, loin de pouvoir remplir ce rôle donc j'ai utilisé **« Sid »** environ 2 ans et on comprends vite le choix du nom de code : c'est imprévisible et violent si l'on est pas prudent.

J'ai aujourd'hui moins de temps à passer à examiner les mises à jour et je roule maintenant en branche ***stable*** . D'abord **« Jessie »** puis **« Stretch »**: les fonctionnalités abordées ici sont

basées sur des paquets très matures qui seront installés sur un système « *headless* » sans fonctions multimédia. Notez que, sauf évolution majeure de ces paquets pourtant stabilisés depuis longtemps, les informations données ici restent valables pour toutes les branches.

Installation

La méthode la plus simple consiste à télécharger l'image de type “Net Install” (installation réseau, ce qui implique qu'il faut une connexion internet active avec un serveur *DHCP*) depuis le site français officiel de Debian puis d'utiliser un outil du type UNetbootin pour transférer l'image sur une clef USB (formatée en FAT32 et pas en NTFS sinon elle ne sera pas *“bootable”*). Il ne reste plus qu'à *“booter”* sur cette clef sur le système de destination et à se laisser guider.

La physionomie du site n'étant pas figée (il y a par ailleurs plusieurs pages menant au bon téléchargement), plutôt que de vous dire de cliquer ici ou là, recherchez une image cd CD au format ISO de type *netinstall* pour architecture AMD64.

J'ai trouvé un échange très intéressant sur le forum Ask Ubuntu mais pour résumer, prendre systématiquement l'image AMD64 à moins d'avoir du matériel ancien limité aux 32bits.

Partitionnement

Après avoir testé différentes possibilités, je suis revenu au plus simple: l'utilisation d'un disque entier pour le système avec une partition principale et une partition Swap.
Pour ce qui est du choix du système de fichiers, on peut être sûr que le choix proposé par défaut par la branche *stable* de Debian est un choix judicieux avec un bon équilibre entre fiabilité et performance, donc, depuis *Jessie*: ext4.

D'autres choix sont évidement possibles mais on ne cherche pas ici à faire une machine de course en recherchant le système de fichier le plus rapide. Pour un serveur, qu'il soit domestique ou pas, il faut privilégier la fiabilité.

Taille du Swap

Que l'on choisisse une partition (ce qui est recommandé étant donné qu'une partition peut profiter d'un système de fichier optimisé pour les opérations de *swap*) ou un simple fichier d'échange, il faut choisir sa taille… Et c'est un sujet brûlant sur les forums  .

Chacun y va de sa formule basée sur la quantité de RAM (généralement entre `RAM*0.5` et `RAM*2`).

Dans les grandes lignes, voici les éléments intéressants que j'ai relevés : - plus la RAM augmente, plus le besoin de *swaper* diminue - à partir de 2 ou 4Go de RAM, 1 ou 2Go de swap est généralement

suffisant - sur un système pour lequel on a prévu d'utiliser l'hibernation, il faut plus de swap que la RAM totale afin qu'elle puisse intégralement être stockée dans le swap au moment de la mise en hibernation (donc de l'avis général RAM+1 devient un minimum réellement requis).

La cas de l'hibernation étant le seul où il y a vraiment un valeur minimale requise objectivement calculable je m'étais orienté vers 5Go pour 4Go de RAM installée mais l'installateur Debian m'a proposé 6Go et qui suis-je pour aller à l'encontre des recommandations la *team* Debian ? 🙂

Choix des fonctionnalités installées

Sélectionner uniquement la ligne “SSH Server” pour obtenir un système minimaliste sans rien de superflu.

Une commande intéressante après coup (une fois que le paquet ''aptitude'' est installé) pour voir la liste des paquets qui se cachent derrière l'élément “Outils standards” (et pas de stress, le commutateur *-s* permet de lancer une simple simulation même si on valide l'installation : <cli>root@muffin:~# aptitude install ~pstandard ~pimportant ~prequired -s Les NOUVEAUX paquets suivants vont être installés :

```
apt-listchanges at bash-completion bc bind9-host bsd-mailx bzip2 dc debian-
faq dnsutils doc-debian
docutils-common{a} docutils-doc{a} exim4 exim4-base exim4-config exim4-
daemon-light fontconfig{a} ftp
geoip-database{a} gnupg-agent{a} gnupg2{a} hicolor-icon-theme{a} host info
install-info iso-codes{a}
libassuan0{a} libatk1.0-0{a} libatk1.0-data{a} libauthen-sasl-perl{a}
libbind9-90 libcairo2{a} libclass-isa-perl
libcurl3-gnutls{a} libdatrie1{a} libdns100 libencode-locale-perl{a}
libev4{a} libevent-2.0-5
libfile-listing-perl{a} libfont-afm-perl{a} libgc1c2 libgdk-pixbuf2.0-0{a}
libgdk-pixbuf2.0-common{a}
libgeoip1{a} libglib2.0-0{a} libglib2.0-data{a} libgpgme11{a}
libgraphite2-3{a} libgssglue1 libgssrpc4
libgtk2.0-0{a} libgtk2.0-bin{a} libgtk2.0-common{a} libharfbuzz0b{a}
libhtml-form-perl{a} libhtml-format-perl{a}
libhtml-parser-perl{a} libhtml-tagset-perl{a} libhtml-tree-perl{a} libhttp-
cookies-perl{a} libhttp-daemon-perl{a}
libhttp-date-perl{a} libhttp-message-perl{a} libhttp-negotiate-perl{a}
libintl-perl{a} libio-html-perl{a}
libio-socket-ssl-perl{a} libisc95 libiscccc90{a} libisccfg90{a} libjasper1{a}
libkadm5clnt-mit9 libkadm5srv-mit9
libkdb5-7 libkrad0 libksba8{a} liblcms2-2{a} liblockfile-bin liblockfile1
liblwp-mediatypes-perl{a}
liblwp-protocol-https-perl{a} liblwres90 libmailtools-perl{a} libnet-http-
perl{a} libnet-smtp-ssl-perl{a}
libnet-ssleay-perl{a} libnfsidmap2 libpango-1.0-0{a} libpangocairo-1.0-0{a}
libpangoft2-1.0-0{a}
libpaper-utils{a} libpaper1{a} libpixman-1-0{a} libpth20{a} librpcsecgss3
```

```
librtmp1{a} libsigsegv2{a} libssh2-1{a}
libswitch-perl libtext-unidecode-perl{a} libthai-data{a} libthai0{a}
libtirpc1 libtokyocabinet9 liburi-perl{a}
libverto-libev1{a} libverto1{a} libwebp5{a} libwebpdemux1{a} libwebpmux1{a}
libwgdb0 libwww-perl{a}
libwww-robotrules-perl{a} libxcb-render0{a} libxcb-shm0{a} libxcomposite1{a}
libxcursor1{a} libxdamage1{a}
libxfixes3{a} libxi6{a} libxinerama1{a} libxml-libxml-perl{a} libxml-
namespacesupport-perl{a}
libxml-parser-perl{a} libxml-sax-base-perl{a} libxml-sax-expat-perl{a}
libxml-sax-perl{a} libxrandr2{a}
libxrender1{a} lsb-release{a} m4 mlocate mutt nfs-common patch pinentry-
gtk2{a} procmail python-apt
python-apt-common{a} python-chardet{a} python-debian{a} python-debianbts{a}
python-defusedxml{a}
python-docutils{a} python-pil{a} python-pkg-resources{a} python-pygments{a}
python-reportbug python-roman{a}
python-six{a} python-soappy{a} python-support python-wstools{a} reportbug
rpcbind shared-mime-info{a} telnet
texinfo time w3m whois xdg-user-dirs{a} xz-utils
```

0 paquets mis à jour, 165 nouvellement installés, 0 à enlever et 0 non mis à jour. Il est nécessaire de télécharger 40,5 Mo d'archives. Après dépaquetage, 151 Mo seront utilisés. Voulez-vous continuer ? [Y/n/?] y Charger/installer/enlever des paquets. root@muffin:~# </cli>

Avec ***Stretch***, j'ai été plus large et j'ai aussi directement installé les “Outils standards” 😎

GRUB

À l'occasion d'une installation qui date déjà un peu, je me suis retrouvé avec un système qui ne pouvais pas booter sans la clef USB utilisée pendant l'installation parce que le système de démarrage *« GRUB »* avait été installé dessus au lieu du disque-dur. Depuis *Wheezhy* ou peut-être même *Squeeze*, l'installateur Debian gère parfaitement les clefs USB.

Mais depuis ***Jessie***, les questions et les explications qui vont avec l'installation du système de démarrage sont assez claires pour qu'il suffise de se laisser guider.

Quelques vérifications

Kernel et mémoire

Il peut être utile de s'assurer que le *kernel* est capable de gérer la mémoire de la machine ou que par exemple toutes les barrettes de RAM fonctionnent : <cli> root@muffin:~# free

```
             total       used       free     shared    buffers     cached
```

Mem: 3758016 1752380 2005636 28564 342820 712584 -/+ buffers/cache: 696976 3061040 Swap: 6369276 0 6369276 </cli> Je n'ai jamais été bon en calcul mental mais là vraiment, toutes ces divisions par 1024 pour avoir une idée précise des valeurs... Donc voyons en Mb : <cli> root@muffin:~# free -m

```
             total       used       free     shared    buffers     cached
```

Mem: 3669 1704 1965 27 334 695 -/+ buffers/cache: 673 2996 Swap: 6219 0 6219 </cli> Voilà qui est bien plus lisible et on retrouve bien les 4Go de RAM installée et les 6 de *swap* (enfin des valeurs suffisament proche pour savoir que tout est OK). La commande *free -g* qui donne les résultats en Gb est vraiment trop approximative pour être exploitable.

Pour la petite histoire, jusqu'à Debian 6 « *Squeeze* », il fallait installer un *kernel* de type “BIGMEM” pour gérer plus de 2Go de RAM.
À partir de « *Wheezy* », il fallait un kernel estampillé *PAE*.
Avec le kernel AMD64, il n'y a pas de telle limite.

Pour vérifier la version du kernel installé : <cli>root@muffin:~# uname -r 3.16.0-4-amd64</cli>

Sources de paquets

<blockquote> Les différents paquets proposés à l’installation sont placés sur des serveurs appelés les « dépôts », et c’est en définissant à quels dépôts votre système aura accès que vous délimiterez la diversité et les versions des programmes disponibles à l’installation sur votre Debian.

Cette liste de dépôts auquel votre système a accès est contenue dans un fichier essentiel à toute Debian : le fichier sources.list (/etc/apt/sources.list), ou dans le répertoire /etc/apt/sources.list.d/ sous formes de plusieurs fichiers. <cite>Wiki du forum debian-fr.org</cite> </blockquote>

Les anciennes version de l'installateur Debian laissaient dans le fichier */etc/apt/sources.list* des traces gênantes du support d'installation (clef USB ou CD) qui empêchaient toute mise à jour ou installation (puisque le système cherchait d'abord à faire son installation à partir du support concerné). Les traces sont toujours là mais de manière inoffensive.

Par ailleurs, pour éviter de télécharger inutilement les listes des sources de paquets disponibles qui ne sont utiles que lorsque l'on souhaite compiler ses propres paquets, il faut aller éditer le fichier */etc/apt/sources.list* et ajouter un # devant chaque ligne commençant par `deb-src`. Voici à quoi il ressemble par défaut avec ***Stretch*** (et personellement, j'ajoute aussi les dépôts `contrib` et `non-free`) :

<cli>root@muffin:/# cp /etc/apt/sources.list /etc/apt/sources.list.original root@muffin:~# vi /etc/apt/sources.list</cli>

sources.list

```
#
```

```
# deb cdrom:[Debian GNU/Linux 9.3.0 _Stretch_ - Official amd64 NETINST
20171209-12:10]/ stretch main

#deb cdrom:[Debian GNU/Linux 9.3.0 _Stretch_ - Official amd64 NETINST
20171209-12:10]/ stretch main

deb http://ftp.fr.debian.org/debian/ stretch main contrib non-free
#deb-src http://ftp.fr.debian.org/debian/ stretch main

deb http://security.debian.org/debian-security stretch/updates main
contrib non-free
#deb-src http://security.debian.org/debian-security stretch/updates
main

# stretch-updates, previously known as 'volatile'
deb http://ftp.fr.debian.org/debian/ stretch-updates main contrib non-
free
#deb-src http://ftp.fr.debian.org/debian/ stretch-updates main
```

Les puristes qui veulent du 100% Open Source devront évidement ne garder que le dépôt `main` mais c'est réellement un choix militant. Un exemple parmi tant d'autres : il sera impossible d'obtenir les firmware des cartes réseau Realtek (le paquet `firmware-realtek` qui est en `non-free`)...

Et voilà!... Le système est prêt...

Minimum vital

Aptitude

L'outil de mise à jour intégré à *Debian* est `apt-get` (ou sans doute plutôt `apt`), mais je lui préfère `aptitude` qui se base dessus mais apporte des fonctions très utile comme la recherche de paquets, l'affichage d'infos pertinente, de version d'un paquet, etc.

Avant un exemple, il faut l'installer : <cli>root@muffin:~# apt-get install aptitude Lecture des listes de paquets... Fait Construction de l'arbre des dépendances Lecture des informations d'état... Fait Paquets suggérés :

```
debtags apt-xapian-index
```

Les NOUVEAUX paquets suivants seront installés :

```
aptitude
```

0 mis à jour, 1 nouvellement installés, 0 à enlever et 0 non mis à jour. Il est nécessaire de prendre 1 506 ko dans les archives. Après cette opération, 4 862 ko d'espace disque supplémentaires seront utilisés. Réception de : 1 http://ftp.fr.debian.org/debian/ jessie/main aptitude amd64 0.6.11-1+b1 [1 506 kB] 1 506 ko réceptionnés en 3s (491 ko/s) Sélection du paquet aptitude précédemment désélectionné. (Lecture de la base de données... 31298 fichiers et répertoires déjà installés.) Préparation du dépaquetage de .../aptitude_0.6.11-1+b1_amd64.deb ... Dépaquetage de aptitude (0.6.11-1+b1) ... Traitement des actions différées (« triggers ») pour man-db (2.7.0.2-5) ... Paramétrage de aptitude (0.6.11-1+b1) ... update-alternatives: utilisation de « /usr/bin/aptitude-curses » pour fournir « /usr/bin/aptitude » (aptitude) en mode automatique </cli>

Aptitude versus apt-get

Prenons l'exemple sur un système doté d'une interface graphique est `Synaptic`. Voici un petit test avec une faute volontaire pour simuler le cas courant où on ne se souviens pas ou on ne connait pas le nom **exact** d'un paquet :

<cli> root@muffin:~# apt-get install synapt Lecture des listes de paquets... Fait Construction de l'arbre des dépendances Lecture des informations d'état... Fait E: Impossible de trouver le paquet synapt root@muffin:~# aptitude install synapt Impossible de trouver le paquet « synapt ». Cependant, les paquets suivants comportent « synapt » dans leur nom :

```
xserver-xorg-input-synaptics-dbg xserver-xorg-input-synaptics xserver-xorg-
input-synaptics-dev
xorg-driver-synaptics synaptic
```

Impossible de trouver le paquet « synapt ». Cependant, les paquets suivants comportent « synapt » dans leur nom :

```
xserver-xorg-input-synaptics-dbg xserver-xorg-input-synaptics xserver-xorg-
input-synaptics-dev
xorg-driver-synaptics synaptic
```

Aucun paquet ne va être installé, mis à jour ou enlevé. 0 paquets mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour. Il est nécessaire de télécharger 0 o d'archives. Après dépaquetage, 0 o seront utilisés. </cli>

Un autre... Le fameux site web, c'est Apache ou Appache ? Bon dans le doute, essayons pache : <cli> root@muffin:~# apt-get install pache Lecture des listes de paquets... Fait Construction de l'arbre des dépendances Lecture des informations d'état... Fait E: Impossible de trouver le paquet pache root@muffin:~# aptitude install pache Impossible de trouver le paquet « pache » et plus de 40 paquets comportent « pache » dans leur nom. Impossible de trouver le paquet « pache » et plus de 40 paquets comportent « pache » dans leur nom. Aucun paquet ne va être installé, mis à jour ou enlevé. 0 paquets mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour. Il est nécessaire de télécharger 0 o d'archives. Après dépaquetage, 0 o seront utilisés. </cli> Avec `apt-get`, on est définitivement dans les cordes alors qu'avec `aptitude` on a une bonne piste (« plus de 40 paquets comportent « pache » dans leur nom ») et le moyen de l'exploiter : <cli> root@muffin:~# aptitude search pache p ampache - système de gestion de fichiers audio pour le web p ampache-

common - système de gestion de fichiers audio pour le web - fichiers co p ampache-themes - thèmes pour Ampache i apache2 - Serveur HTTP Apache v apache2-api-20120211 - i A apache2-bin - Serveur HTTP Apache (modules et autres fichiers binaires) i A apache2-data - Serveur HTTP Apache - fichiers communs p apache2-dbg - symboles de débogage pour Apache p apache2-dev - Apache HTTP Server (development headers) .../... p rhythmbox-ampache - play audio streams from an Ampache server p rt4-apache2 - Apache 2 specific files for request-tracker4 v torrus-apache2 - </cli>

Et `aptitude` a de nombreuses autres possibilités et il suffit de cette commande pour s'en convaincre :

<cli>root@muffin:~# aptitude -h</cli>

Voici maintenant, à mes yeux, le seul point sur lequel `apt-get` bat `aptitude` à plate couture : <cli>root@muffin:~# apt-get –help | grep -i cow

```
Cet APT a les « Super Cow Powers »
```

root@muffin:~# aptitude –help | grep -i cow

```
Cet aptitude n'a pas de « Super Cow Powers ».
```

</cli>

Je vous invite à creuser la question en faisant une petite recherche sur la toile 🤣.

Vim

Certains se contenteront de *vi* mais je préfère son petit frère:

<cli>root@muffin:~# aptitude install vim</cli>

Activer ensuite la coloration syntaxique avec le thème “désert” qui utilise des couleurs plus lisibles (il faut rechercher la première ligne et ajouter la seconde): <cli>root@muffin:/# cp /etc/vim/vimrc /etc/vim/vimrc.original root@muffin:~# vi /etc/vim/vimrc</cli> Il faut décommenter la ligne `syntax on` et ajouter la ligne `colorscheme` :

```
.../...
syntax on
colorscheme desert
.../...
```

A partir de Stretch

Par défaut, un clic-droit ne colle plus le contenu du presse-papier mais bascule en mode *Visual*. Pour changer ce comportement par défaut, il suffit normalement de créer un fichier `~/.vimrc` vide (ou, s'il existe déjà, d'y ajouter une ligne `set mouse-=a`).

Première mise à jour

Normalement, le système est parfaitement à jour puisque nous sommes passés par une installation depuis internet mais les commandes qui suivent vont permettre de s'assurer que le système fonctionne correctement et accède bien à internet: <cli>root@muffin:~# aptitude update && aptitude upgrade</cli>

Paquets recommandés

Parfois, lors d'une mise à jour, `aptitude` indique une liste de paquets recommandés. Il est évidement **fortement** conseillé de jeter systématiquement un œil à cette éventuelle liste.

<cli>Les paquets suivants seront mis à jour :

```
base-files dbus libdbus-1-3 libgnutls-deb0-28 libgnutls-openssl27
libsystemd0 libudev1 linux-image-3.16.0-4-amd64
perl perl-base perl-modules systemd systemd-sysv task-french task-ssh-server
tasksel tasksel-data udev
```

Les paquets suivants sont RECOMMANDÉS mais ne seront pas installés :

```
firmware-linux-free irqbalance libpam-systemd
```

18 paquets mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour. Il est nécessaire de télécharger 45,3 Mo d'archives. Après dépaquetage, 45,1 ko seront utilisés. Voulez-vous continuer ? [Y/n/?] y </cli>

Mais ils ne sont pas indispensables... Donc il faut vérifier le descriptif pour ne pas installer n'importe quoi.

Dans l'exemple ci-dessus, c'est toujours une bonne idée d'installer le paquet `firmware-linux-free` qui contiens tous les *firmwares open source* possibles. Le descriptif de `irqbalance` indique une meilleure gestion des IRQ avec les processeurs *multi-cores* ce qui est le cas de celui qui équipe mon serveur. Par contre, j'attends d'en savoir plus sur *SystemD* avant d'installer `libpam-systemd`.

À la recherche du fichier perdu

Au moment d'une mise à jour, Debian lance parfois des messages un peu déroutants à propos de firmware(s) manquant(s) alors que tout semble fonctionner...

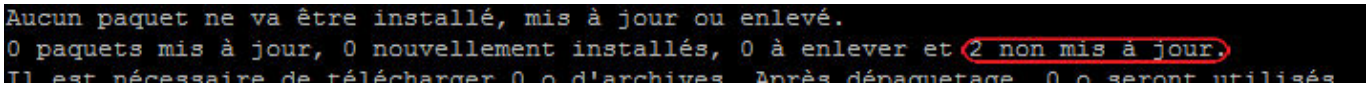
Ici par exemple un problème de firmware de carte réseau sur une ancienne carte-mère (alors que l'accès réseau fonctionnait) : <cli>update-initramfs: Generating /boot/initrd.img-3.2.0-4-686-pae W: Possible missing firmware /lib/firmware/rtl_nic/rtl8168f-2.fw for module r8169 W: Possible missing firmware /lib/firmware/rtl_nic/rtl8168f-1.fw for module r8169 .../... </cli> Le problème est donc de trouver quel paquet installer pour obtenir le ou les fichiers manquants et c'est exactement le rôle de `apt-file`. <cli>root@muffin:~# aptitude install apt-file</cli> Ce paquet à besoin d'une liste à jour des fichiers existants qu'il faut mettre à jour à partir des dépôts actifs avant chaque utilisation :

<cli>root@muffin:~# apt-file update</cli> Il ne reste plus qu'à rechercher le paquet nécessaire, dans mon cas: <cli>root@muffin:~# apt-file search rtl8168f-2.fw firmware-realtek: /lib/firmware/rtl_nic/rtl8168f-2.fw</cli> Le paquet qui manque ici et qu'il suffit d'installer est donc *firmware-realtek*.

Dans ce cas précis, si le dépôt “non-free” n'avait pas été ajouté au fichier *sources.list* avant la mise à jour de la base de données d'*apt-file* la commande n'aurait renvoyé aucun résultat.

Paquets non mis à jour

La commande `aptitude upgrade` indique parfois que certains paquets nécessitent une attention particulière pour être mis à jour et ne seront pas modifiés:



Pour mettre à jour ces paquets, il faut utiliser cette commande: <cli>root@muffin:~# aptitude full-upgrade</cli> <blockquote>.../...
full-upgrade : mettre à jour les paquets dans leur version la plus récente, en supprimant ou installant autant de paquets que nécessaire. Cette commande est moins conservatrice que safe-upgrade et donc a plus de chance de provoquer des actions inattendues.
.../... <cite>man aptitude</cite> </blockquote>

la commande *aptitude dist-upgrade* est rigoureusement identique mais tout simplement plus ancienne

Un peu de couleurs et alias utiles

Par défaut, la console de l'utilisateur *root* est très austère avec tout ce texte blanc sur fond noir. On peut aller très très loin dans la colorisation du prompt (au point de piquer les yeux 😬) et il n'est pas question de cela ici, par contre la colorisation des résultats des commandes `ls` est indispensable (et d'ailleurs prévue puisqu'il suffit de décommenter quelques ligne d'un fichier) pour distinguer rapidement les fichiers des dossiers et de repérer les liens symboliques.

Les alias permettent de modifier les options par défaut apliquées avec certaines commandes. C'est particulièrement utile en tant que *root* pour éviter de commettre des bêtises avec la commande `rm` : <cli>root@muffin:~# cp /root/.bashrc /root/.bashrc.original root@muffin:~# vi /root/.bashrc</cli>

.bashrc

```
# ~/.bashrc: executed by bash(1) for non-login shells.
```

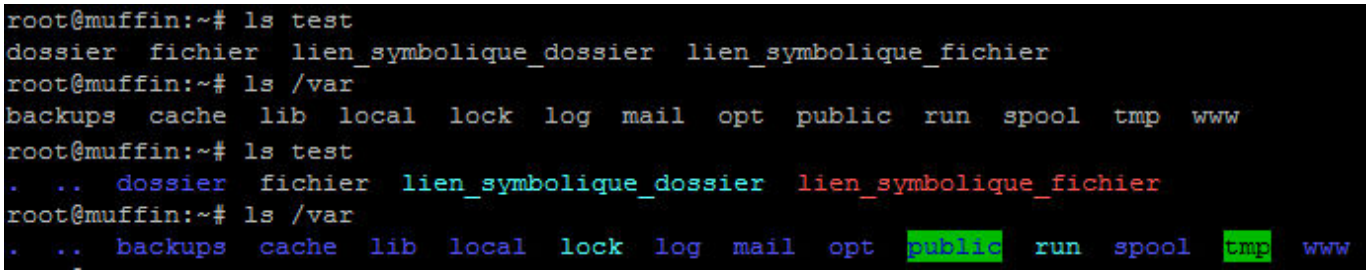
```
# Note: PS1 and umask are already set in /etc/profile. You should not
# need this unless you want different defaults for root.
# PS1='${debian_chroot:+($debian_chroot)}\h:\w\$ '
# umask 022

# You may uncomment the following lines if you want `ls' to be
colorized:
 export LS_OPTIONS='--color=auto -a'
 eval "`dircolors`"
 alias ls='ls $LS_OPTIONS'
 alias ll='ls $LS_OPTIONS -l'
 alias l='ls $LS_OPTIONS -lA'

# Some more alias to avoid making mistakes:
 alias rm='rm -i'
# alias cp='cp -i'
# alias mv='mv -i'
```

notez que la ligne *export LS_OPTIONS...* est modifiée pour ajouter systématiquement le commutateur `-a` en plus d'être décommentée

Et voici un petit exemple comparatif en image :



Scripts et variable PATH

L'idée est de choisir où placer tous les petits scripts (ou les gros) que tout administrateur système est amené à écrire... La Filesystem_Hierarchy_Standard (*Filesystem Hierarchy Standard*) dis ceci :

/usr/local/sbin ⇒ custom script meant for root
/usr/local/bin ⇒ custom script meant for all users including non-root

Et Debian précise que le dossier `/usr/local` « est réservé à l'usage privé de l'administrateur

système ».

Malheureusement quelques paquets ne respectent pas ces conventions et s'installent dans ces répertoires. Donc plutôt que de devoir chercher mes scripts aux milieu de tout un fatras de fichiers, je préfère les isoler à un endroit dont je suis le seul à avoir la maîtrise. Le seul problème que cela pose (en dehors du risque de se faire traiter d'iconoclaste par un psychorigide des conventions 😬) est qu'il faut ajouter le chemin en question à la variable d'environnement *PATH* afin de pouvoir les lancer sans taper le chemin complet.

On peut modifier cette variable d'environnement pour tous les utilisateurs qui se connectent au système en modifiant le fichier */etc/bash.bashrc* ou uniquement pour le *root* en modifiant le fichier */root/.bashrc*. Il suffit d'y ajouter la ligne suivante (dans les deux cas juste sous la première ligne de commentaire par exemple) :

```
export PATH="$PATH:/mon/chemin/scripts"
```

Attention : si le chemin n'est pas sur le disque système, les scripts ne seront pas forcément disponibles très tôt pendant le démarrage du système. Ce n'est pas forcément problématique mais c'est à prendre en considération

Il ne lui manque que la voix

Avant de commencer

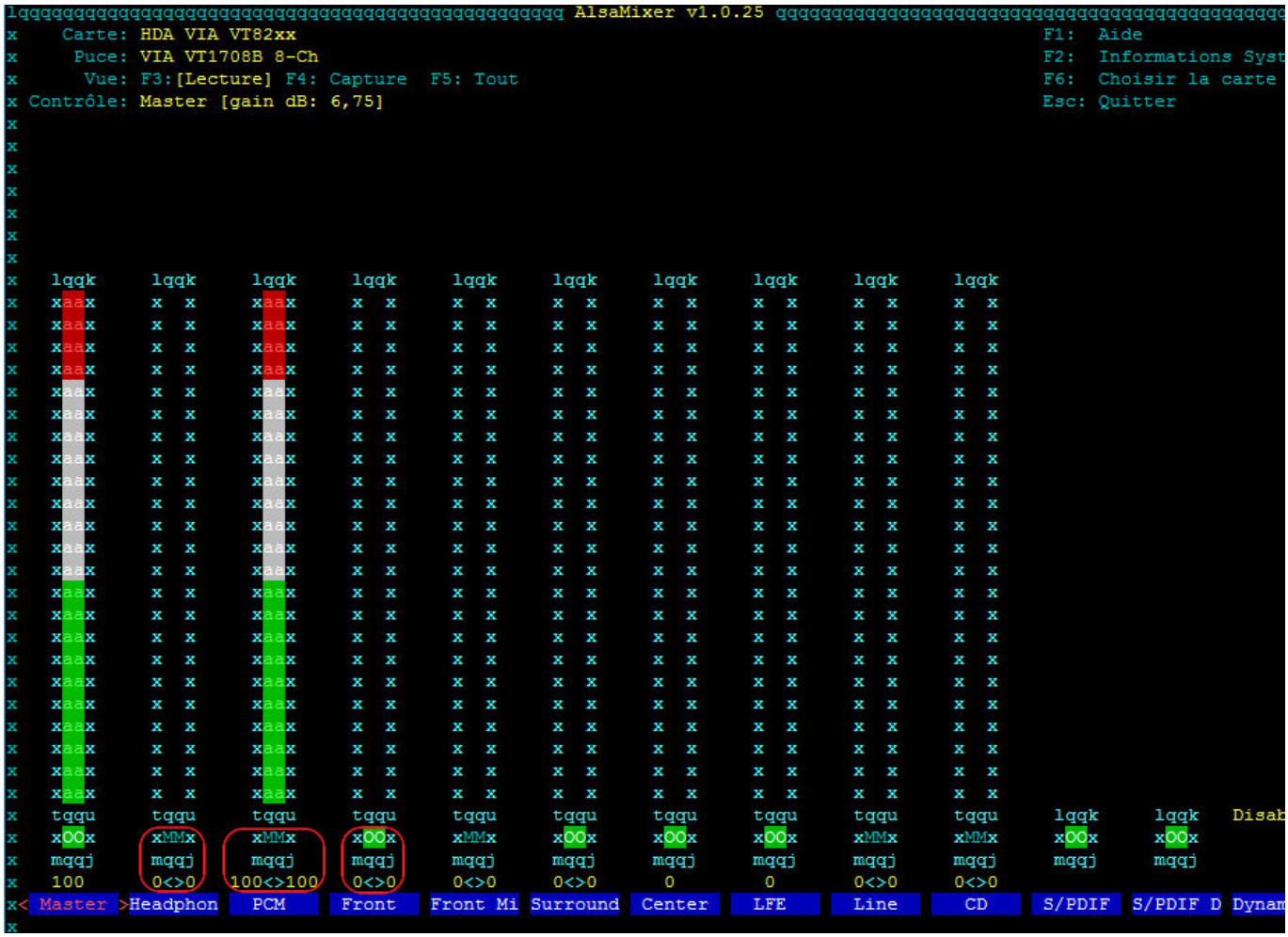
Contrairement à ce que l'on pourrait penser, il est bien utile de pouvoir émettre des sons sur un serveur « *headless* » lors de certains événements, ne serait-ce que pour signaler la fin d'une procédure de démarrage.
Comme la reconnaissance du hardware est une faiblesse de Linux, il faut impérativement s'assurer que le système est à jour pour diminuer le risque de problèmes: <cli>root@muffin:~# aptitude update && aptitude upgrade</cli>

C'est parti

<cli>root@muffin:~# aptitude install alsa-utils</cli>

Avec **Jessie**, il fallait aussi installer le paquet `alsa-base` mais il n'existe plus et cela fonction sans...

Par défaut, le volume de certaines voix audio (ou de toutes), peut être au minimal ou même coupé et on ne risque pas d'entendre quoi que ce soit. <cli>root@muffin:~# alsamixer</cli>



Ici, la voix “Headphone” est en mode *mute* avec un volume à 0, la voix “PCM” est en mode *mute* avec un volume à 100, la voix “Front” est en mode normal mais son volume est à 0, ... Dans un premier temps, le plus simple est de passer sur chaque voix pour mettre le volume au maximum (avec les touches directionnelles du clavier), et, le cas échéant, de désactiver le mode *mute* (avec la touche [M] une fois la voix concernée sélectionnée. Une fois que c'est fait, la touche [ESCAPE] permet de quitter l'égaliseur et il ne reste qu'à tester: <cli>root@muffin:~# speaker-test</cli> Si un ignoble crachotement se fait entendre, c'est gagné ! 😳
On peut retourner dans l'égaliseur pour faire quelques tests et désactiver les voix inutiles (avec l'ancienne carte mère Jetway comme avec la nouvelle ASRock, les voix utiles sont: “Master”, “PCM” et “Front”).
Un autre petit test (nettement moins désagréable pour les oreilles) consiste à jouer un fichier audio : <cli>root@muffin:~# aplay /chemin/vers/une/source/fichier.wav</cli>